Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №1 по курсу

«Операционные системы»

Группа: М8О-213Б-23

Студент: Германенко М. И.

Преподаватель: Бахарев В.Д.

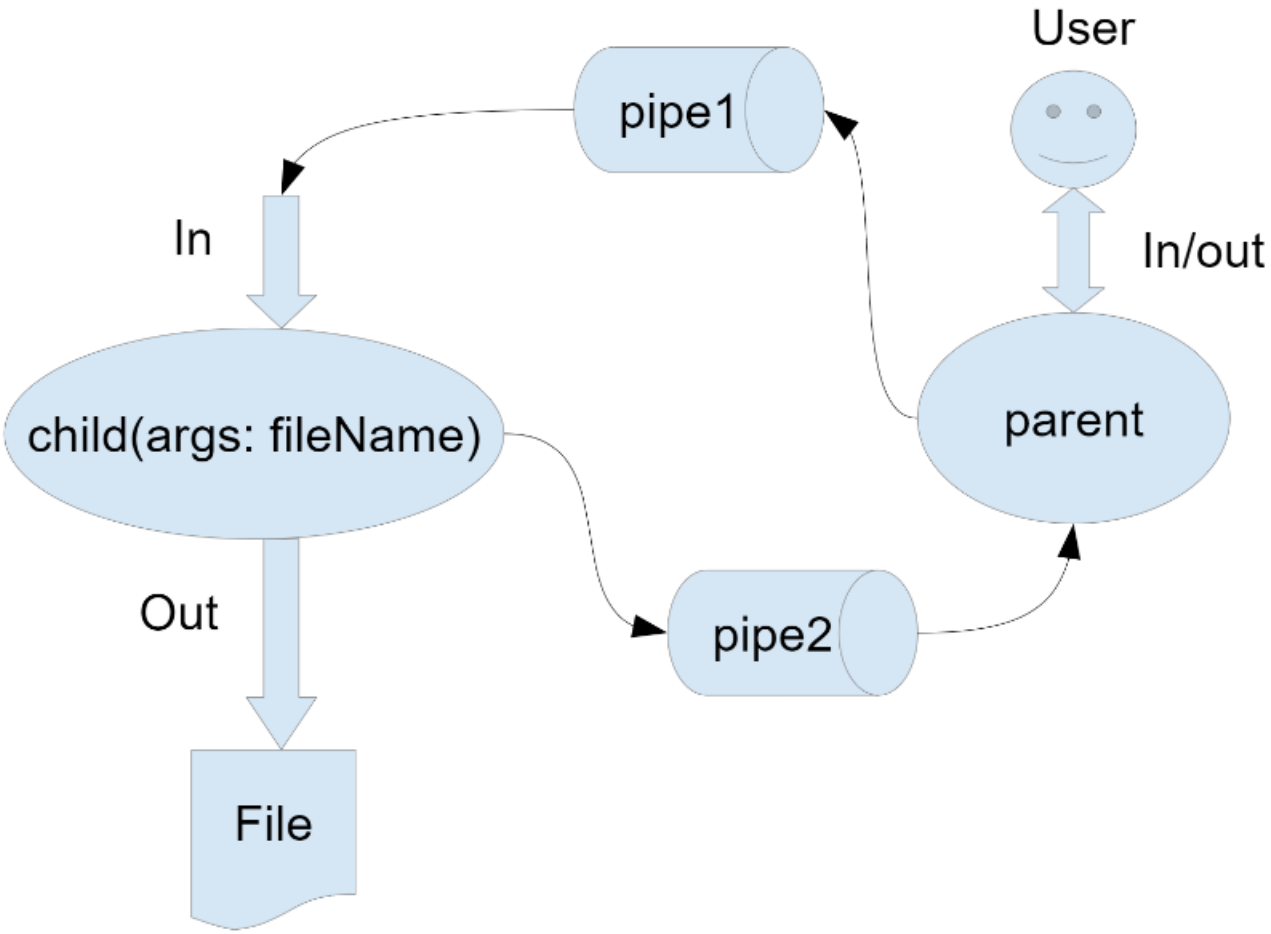
Оценка: _____

Дата: 17.10.24

Москва, 2024

Постановка задачи

Вариант 5.



Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс принеобходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

5 вариант) Пользователь вводит команды вида: «число<endline>». Далее это число передается от родительского процесса в дочерний. Дочерний процесс производит проверку на простоту. Если число составное, то в это число записывается в файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pipe(int pipefd[2]) - создает канал (pipe) для межпроцессного взаимодействия, возвращая два файловых дескриптора: один для чтения (pipefd[0]), другой для записи (pipefd[1]).
- fork() - создает новый процесс (дочерний процесс) путем копирования родительского. В родительском процессе возвращает PID дочернего процесса, в дочернем — 0.
- read(int fd, void *buf, size_t count) - читает данные из файлового дескриптора fd в буфер buf. Возвращает количество прочитанных байтов.
- write(int fd, const void *buf, size_t count) - записывает данные из буфера buf в файловый дескриптор fd. Возвращает количество записанных байтов.

- dup2(int oldfd, int newfd) - дублирует файловый дескриптор oldfd на дескриптор newfd, заменяя последний.
- execl(const char *path, const char *arg, ..., NULL) - запускает новый процесс, заменяя текущий образ программы процессом, находящимся по пути path. Аргументы передаются через список аргументов (например, имя файла).
- open(const char *pathname, int flags, mode_t mode) - открывает файл по пути pathname с указанными флагами и режимом (например, создание нового файла). Возвращает файловый дескриптор.
- close(int fd) - закрывает файловый дескриптор, освобождая ресурсы.
- (int *status) - ожидает завершения дочернего процесса и возвращает его статус.
- exit(int status) - завершает выполнение текущего процесса, возвращая статус завершения.
- fsync(int fd) – синхронизирует данные, записанные в файловый дескриптор fd, с диском, гарантируя физическую запись данных.

Родительский процесс создаёт два канала (pipe) для обмена данными с дочерним процессом: pipe1 используется для передачи чисел от родительского к дочернему процессу. pipe2 используется для передачи сигналов завершения от дочернего к родительскому процессу. Родительский процесс запрашивает у пользователя имя файла для сохранения данных. Имя файла передаётся дочернему процессу через аргумент при запуске программы дочернего процесса с помощью системного вызова execl. Вызов fork() создает два процесса:

Родительский процесс:

- Закрывает ненужные части каналов.
- Принимает числовой ввод от пользователя, проверяет его на корректность и передаёт числа дочернему процессу через pipe1.
- Ожидает сигнал от дочернего процесса через pipe2, чтобы понять, завершить ли программу.

Дочерний процесс:

- Получает имя файла в качестве аргумента через execl.
- Открывает указанный файл для записи чисел.
- Читает числа из pipe1 и записывает их в файл, разделяя пробелами.
- Проверяет, является ли число простым, и если оно простое или отрицательное, завершает выполнение, отправляя сигнал родительскому процессу через pipe2.

После получения числа от пользователя, родительский процесс передаёт его дочернему через канал pipe1. Дочерний процесс принимает числа, проверяет их, и записывает в файл через системные вызовы write, разделяя числа пробелами. Если число простое или отрицательное, процесс завершается. Дочерний процесс отправляет сигнал завершения (значение 1) через pipe2, если полученное число является простым или отрицательным, что заставляет родительский процесс завершить выполнение программы.

Код программы

parent.c

```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>
#include <fcntl.h>

void handle_error(const char *message) {
```

```
    write(STDERR_FILENO, message, strlen(message));
    exit(EXIT_FAILURE);
}

int main() {
    int pipe1[2], pipe2[2];

    if (pipe(pipe1) == -1) {
        handle_error("Ошибка при создании pipe1\n");
    }

    if (pipe(pipe2) == -1) {
        handle_error("Ошибка при создании pipe2\n");
    }

    char filename[128];

    write(STDOUT_FILENO, "Введите имя файла: ", 34);
    fsync(STDOUT_FILENO);
    int n = read(STDIN_FILENO, filename, sizeof(filename));

    if (n <= 0) {
        handle_error("Ошибка при чтении имени файла\n");
    }
    filename[n - 1] = '\0';

    pid_t pid = fork();

    if (pid > 0) {
        close(pipe1[0]);
        close(pipe2[1]);

        char input[128];
        int number, signal;

        while (1) {
            write(STDOUT_FILENO, "Введите число: ", 28);
            fsync(STDOUT_FILENO);
            int n = read(STDIN_FILENO, input, sizeof(input));

            if (n <= 0) {
                handle_error("Ошибка при чтении числа\n");
            }

            input[n - 1] = '\0';

            char *endptr;
            number = strtol(input, &endptr, 10);

            if (endptr == input || *endptr != '\0') {
                handle_error("Ошибка: вводите только числа\n");
            }

            if (write(pipe1[1], &number, sizeof(number)) == -1) {
```

```
                handle_error("Ошибка при записи в pipe1\n");
            }

            if (read(pipe2[0], &signal, sizeof(signal)) == -1) {
                handle_error("Ошибка при чтении из pipe2\n");
            }

            if (signal == 1) {
                write(STDOUT_FILENO, "Программа завершена.\n", 40);
                break;
            }
        }

        close(pipe1[1]);
        close(pipe2[0]);

        if (wait(NULL) == -1) {
            handle_error("Ошибка при ожидании дочернего процесса\n");
        }
    } else if (pid == 0) {
        close(pipe1[1]);
        close(pipe2[0]);

        if (dup2(pipe1[0], STDIN_FILENO) == -1) {
            handle_error("Ошибка при перенаправлении pipe1 в stdin\n");
        }

        if (dup2(pipe2[1], STDOUT_FILENO) == -1) {
            handle_error("Ошибка при перенаправлении pipe2 в stdout\n");
        }

        execl("./child", "./child", filename, NULL);

        handle_error("Ошибка при запуске дочернего процесса\n");
    } else {
        handle_error("Ошибка при вызове fork\n");
    }

    return 0;
}
```

child.c

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>

void handle_error(const char *message) {
    write(STDERR_FILENO, message, strlen(message));
    exit(EXIT_FAILURE);
}

int is_prime(int num) {
    if (num < 2) {
```

```c
        return 0;
    }
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return 0;
        }
    }
    return 1;
}

int int_to_str(int num, char *buf, int buf_size) {
    int len = 0;
    int temp = num;

    if (num < 0) {
        if (buf_size > 1) {
            buf[len++] = '-';
            num = -num;
        } else {
            return -1;
        }
    }

    do {
        temp /= 10;
        len++;
    } while (temp > 0);

    if (len >= buf_size) {
        return -1;
    }

    buf[len] = '\0';
    while (num > 0) {
        buf[--len] = (num % 10) + '0';
        num /= 10;
    }

    return strlen(buf);
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        handle_error("Ошибка: имя файла не передано\n");
    }

    const char *filename = argv[1];
    int number, signal;

    int file_fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0666);
    if (file_fd == -1) {
        handle_error("Ошибка при открытии файла\n");
    }
```

```
    while (1) {
        if (read(STDIN_FILENO, &number, sizeof(number)) <= 0) {
            handle_error("Ошибка при чтении из pipe1\n");
        }

        if (number < 0 || is_prime(number)) {
            signal = 1;
            if (write(STDOUT_FILENO, &signal, sizeof(signal)) == -1) {
                handle_error("Ошибка при записи сигнала в pipe2\n");
            }
            break;
        } else {
            char buf[128];
            int len = int_to_str(number, buf, sizeof(buf));
            if (len == -1) {
                handle_error("Ошибка при преобразовании числа в строку\n");
            }

            if (write(file_fd, buf, len) == -1) {
                handle_error("Ошибка при записи в файл\n");
            }

            if (write(file_fd, " ", 1) == -1) {
                handle_error("Ошибка при записи пробела в файл\n");
            }

            signal = 0;
            if (write(STDOUT_FILENO, &signal, sizeof(signal)) == -1) {
                handle_error("Ошибка при записи сигнала в pipe2\n");
            }
        }
    }

    if (close(file_fd) == -1) {
        handle_error("Ошибка при закрытии файла\n");
    }

    exit(0);
}
```

Протокол работы программы

Тестирование:

migermanenko@MacBook-Pro-Matvej src % ./parent

Введите имя файла: test.txt

Введите число: 24

Введите число: 88

Введите число: 2f

Ошибка: вводите только числа

Ошибка при чтении из pipe1

migermanenko@MacBook-Pro-Matvej src % ./parent

Введите имя файла: test.txt

Введите число: 24

Введите число: 56

Введите число: 1

Введите число: 2

Программа завершена.

migermanenko@MacBook-Pro-Matvej src % ./parent

Введите имя файла: test.txt

Введите число: 40

Введите число: -40

Программа завершена.

Dtruss:

migermanenko@MacBook-Pro-Matvej src % sudo dtruss -f ./parent

PID/THRD SYSCALL(args) = return

Введите имя файла: 1116/0x2dc1: fork() = 0 0

1116/0x2dc1: munmap(0x100FA4000, 0x84000) = 0 0

1116/0x2dc1: munmap(0x101028000, 0x8000) = 0 0

1116/0x2dc1: munmap(0x101030000, 0x4000) = 0 0

1116/0x2dc1: munmap(0x101034000, 0x4000) = 0 0

1116/0x2dc1: munmap(0x101038000, 0x48000) = 0 0

1116/0x2dc1: munmap(0x101080000, 0x4C000) = 0 0

1116/0x2dc1: crossarch_trap(0x0, 0x0, 0x0) = -1 Err#45

1116/0x2dc1: open(".\0", 0x100000, 0x0) = 3 0

1116/0x2dc1: fcntl(0x3, 0x32, 0x16F2070C8) = 0 0

1116/0x2dc1: close(0x3) = 0 0

1116/0x2dc1: fsgetpath(0x16F2070D8, 0x400, 0x16F2070B8) = 51 0

1116/0x2dc1: fsgetpath(0x16F2070E8, 0x400, 0x16F2070C8) = 14 0

1116/0x2dc1: csrctl(0x0, 0x16F2074EC, 0x4) = -1 Err#1

1116/0x2dc1: __mac_syscall(0x18CA1FC12, 0x2, 0x16F207430) = 0 0

1116/0x2dc1: csrctl(0x0, 0x16F2074DC, 0x4) = -1 Err#1

1116/0x2dc1: __mac_syscall(0x18CA1CA45, 0x5A, 0x16F207470) = 0 0

1116/0x2dc1: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16F2069D8, 0x16F2069D0, 0x18CA1E738, 0xD) = 0 0

1116/0x2dc1: sysctl([CTL_KERN, 157, 0, 0, 0, 0] (2), 0x16F206A88, 0x16F206A80, 0x0, 0x0) = 0 0

1116/0x2dc1: open("/\0", 0x20100000, 0x0) = 3 0

1116/0x2dc1: openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0) = 4 0

1116/0x2dc1: dup(0x4, 0x0, 0x0) = 5 0

1116/0x2dc1: fstatat64(0x4, 0x16F206561, 0x16F2064D0) = 0 0

1116/0x2dc1: openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0) = 6 0

1116/0x2dc1: fcntl(0x6, 0x32, 0x16F206560) = 0 0

1116/0x2dc1: dup(0x6, 0x0, 0x0) = 7 0

1116/0x2dc1: dup(0x5, 0x0, 0x0) = 8 0

1116/0x2dc1: close(0x3) = 0 0

1116/0x2dc1: close(0x5) = 0 0

1116/0x2dc1: close(0x4) = 0 0

1116/0x2dc1: close(0x6) = 0 0

1116/0x2dc1: __mac_syscall(0x18CA1FC12, 0x2, 0x16F206F50) = 0 0

1116/0x2dc1: shared_region_check_np(0x16F206B70, 0x0, 0x0) = 0 0

1116/0x2dc1: fsgetpath(0x16F2070F0, 0x400, 0x16F207018) = 82 0

1116/0x2dc1: fcntl(0x8, 0x32, 0x16F2070F0) = 0 0

1116/0x2dc1: close(0x8) = 0 0

1116/0x2dc1: close(0x7) = 0 0

1116/0x2dc1: getfsstat64(0x0, 0x0, 0x2) = 11 0

1116/0x2dc1: getfsstat64(0x100BF4050, 0x5D28, 0x2) = 11 0

1116/0x2dc1: getattrlist("/\0", 0x16F207030, 0x16F206FA0) = 0 0

1116/0x2dc1: stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld_shared_cache_arm64e\0", 0x16F207390, 0x0) = 0 0

dtrace: error on enabled probe ID 1696 (ID 845: syscall::stat64:return): invalid address (0x0) in action #12 at DIF offset 12

1116/0x2dc1: stat64("/Users/migermanenko/Documents/C/OC/Lab1/src/parent\0", 0x16F206840, 0x0) = 0 0

1116/0x2dc1: open("/Users/migermanenko/Documents/C/OC/Lab1/src/parent\0", 0x0, 0x0) = 3 0

1116/0x2dc1: mmap(0x0, 0x8508, 0x1, 0x40002, 0x3, 0x0) = 0x100BF4000 0

1116/0x2dc1: fcntl(0x3, 0x32, 0x16F206958) = 0 0

1116/0x2dc1: close(0x3) = 0 0

1116/0x2dc1: munmap(0x100BF4000, 0x8508) = 0 0

1116/0x2dc1: stat64("/Users/migermanenko/Documents/C/OC/Lab1/src/parent\0", 0x16F206DB0, 0x0) = 0 0

1116/0x2dc1: stat64("/usr/lib/libSystem.B.dylib\0", 0x16F205D00, 0x0) = -1 Err#2

1116/0x2dc1: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16F205CB0, 0x0) = -1 Err#2

1116/0x2dc1: open("/Users/migermanenko/Documents/C/OC/Lab1/src/parent\0", 0x0, 0x0) = 3 0

1116/0x2dc1: __mac_syscall(0x18CA1FC12, 0x2, 0x16F204400) = 0 0

1116/0x2dc1: map_with_linking_np(0x16F2042B0, 0x1, 0x16F2042E0) = 0 0

1116/0x2dc1: close(0x3) = 0 0

1116/0x2dc1: mprotect(0x100BEC000, 0x4000, 0x1) = 0 0

1116/0x2dc1: open("/dev/dtracehelper\0", 0x2, 0x0) = 3 0

1116/0x2dc1: ioctl(0x3, 0x80086804, 0x16F2039B8) = 0 0

1116/0x2dc1: close(0x3) = 0 0

1116/0x2dc1: shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

1116/0x2dc1: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2

1116/0x2dc1: bsdthread_register(0x18CD220F4, 0x18CD220E8, 0x4000) = 1073746399 0

1116/0x2dc1: getpid(0x0, 0x0, 0x0) = 1116 0

1116/0x2dc1: shm_open(0x18CBB9F41, 0x0, 0xFFFFFFFF8CD60000) = 3 0

1116/0x2dc1: fstat64(0x3, 0x16F204030, 0x0) = 0 0

1116/0x2dc1: mmap(0x0, 0x8000, 0x1, 0x40001, 0x3, 0x0) = 0x100BFC000 0

1116/0x2dc1: close(0x3) = 0 0

1116/0x2dc1: csops(0x45C, 0x0, 0x16F20416C) = 0 0

1116/0x2dc1: ioctl(0x2, 0x4004667A, 0x16F2040DC) = 0 0

1116/0x2dc1: mprotect(0x100C0C000, 0x4000, 0x0) = 0 0

1116/0x2dc1: mprotect(0x100C18000, 0x4000, 0x0) = 0 0

1116/0x2dc1: mprotect(0x100C1C000, 0x4000, 0x0) = 0 0

1116/0x2dc1: mprotect(0x100C28000, 0x4000, 0x0) = 0 0

1116/0x2dc1: mprotect(0x100C2C000, 0x4000, 0x0) = 0 0

1116/0x2dc1: mprotect(0x100C38000, 0x4000, 0x0) = 0 0

1116/0x2dc1: mprotect(0x100C04000, 0xC8, 0x1) = 0 0

1116/0x2dc1: mprotect(0x100C04000, 0xC8, 0x3) = 0 0

1116/0x2dc1: mprotect(0x100C04000, 0xC8, 0x1) = 0 0

1116/0x2dc1: mprotect(0x100C3C000, 0x4000, 0x1) = 0 0

1116/0x2dc1: mprotect(0x100C40000, 0xC8, 0x1) = 0 0

1116/0x2dc1: mprotect(0x100C40000, 0xC8, 0x3) = 0 0

1116/0x2dc1: mprotect(0x100C40000, 0xC8, 0x1) = 0 0

1116/0x2dc1: mprotect(0x100C04000, 0xC8, 0x3) = 0 0

1116/0x2dc1: mprotect(0x100C04000, 0xC8, 0x1) = 0 0

1116/0x2dc1: mprotect(0x100C3C000, 0x4000, 0x3) = 0 0

1116/0x2dc1: mprotect(0x100C3C000, 0x4000, 0x1) = 0 0

1116/0x2dc1: issetugid(0x0, 0x0, 0x0) = 0 0

1116/0x2dc1: getentropy(0x16F203738, 0x20, 0x0) = 0 0

1116/0x2dc1: getattrlist("/Users/migermanenko/Documents/C/OC/Lab1/src/parent\0", 0x16F203FD0, 0x16F203FEC) = 0 0

1116/0x2dc1: access("/Users/migermanenko/Documents/C/OC/Lab1/src\0", 0x4, 0x0) = 0 0

1116/0x2dc1: open("/Users/migermanenko/Documents/C/OC/Lab1/src\0", 0x0, 0x0) = 3 0

1116/0x2dc1: fstat64(0x3, 0x129E04500, 0x0) = 0 0

1116/0x2dc1: csrctl(0x0, 0x16F2041BC, 0x4) = 0 0

1116/0x2dc1: fcntl(0x3, 0x32, 0x16F203EB8) = 0 0

1116/0x2dc1: close(0x3) = 0 0

1116/0x2dc1: open("/Users/migermanenko/Documents/C/OC/Lab1/src/Info.plist\0", 0x0, 0x0) = -1 Err#2

1116/0x2dc1: proc_info(0x2, 0x45C, 0xD) = 64 0

1116/0x2dc1: csops_audittoken(0x45C, 0x10, 0x16F204240) = 0 0

1116/0x2dc1: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16F204598, 0x16F204590, 0x19044CD3A, 0x15) = 0 0

1116/0x2dc1: sysctl([CTL_KERN, 155, 0, 0, 0, 0] (2), 0x16F204628, 0x16F204620, 0x0, 0x0) = 0 0

1116/0x2dc1: pipe(0x0, 0x0, 0x0) = 3 0

1116/0x2dc1: pipe(0x0, 0x0, 0x0) = 5 0

1116/0x2dc1: write(0x1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260: \0", 0x22) = 34 0

1116/0x2dc1: fsync(0x1, 0x0, 0x0) = 0 0

test.txt

Введите число: 1116/0x2dc1: read(0x0, "test.txt\n\0", 0x80) = 9 0

1116/0x2dc1: fork() = 1131 0

1131/0x2e83: fork() = 0 0

1131/0x2e83: thread_selfid(0x0, 0x0, 0x0) = 11907 0

1131/0x2e83: bsdthread_register(0x18CD220F4, 0x18CD220E8, 0x4000) = -1 Err#22

1116/0x2dc1: close(0x3) = 0 0

1116/0x2dc1: close(0x6) = 0 0

1131/0x2e83: mprotect(0x100C40000, 0xC8, 0x3) = 0 0

1116/0x2dc1: write(0x1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\276: \0", 0x1C) = 28 0

1131/0x2e83: mprotect(0x100C40000, 0xC8, 0x1) = 0 0

1116/0x2dc1: fsync(0x1, 0x0, 0x0) = 0 0

1131/0x2e83: close(0x4) = 0 0

1131/0x2e83: close(0x5) = 0 0

1131/0x2e83: dup2(0x3, 0x0, 0x0) = 0 0

1131/0x2e83: dup2(0x6, 0x1, 0x0) = 1 0

dtrace: error on enabled probe ID 1694 (ID 287: syscall::execve:return): invalid address
(0x100bebf27) in action #12 at DIF offset 12

1131/0x2e84: fork() = 0 0

1131/0x2e84: mprotect(0x1010C0000, 0x8000, 0x1) = 0 0

1131/0x2e84: thread_selfid(0x0, 0x0, 0x0) = 11908 0

1131/0x2e84: crossarch_trap(0x0, 0x0, 0x0) = -1 Err#45

1131/0x2e84: shared_region_check_np(0x16EE03780, 0x0, 0x0) = 0 0

1131/0x2e84: thread_selfid(0x0, 0x0, 0x0) = 11908 0

1131/0x2e84: getpid(0x0, 0x0, 0x0) = 1131 0

1131/0x2e84: proc_info(0xF, 0x46B, 0x0) = 0 0

1131/0x2e84: munmap(0x10103C000, 0x84000) = 0 0

1131/0x2e84: munmap(0x1010C0000, 0x8000) = 0 0

1131/0x2e84: munmap(0x1010C8000, 0x4000) = 0 0

1131/0x2e84: munmap(0x1010CC000, 0x4000) = 0 0

1131/0x2e84: munmap(0x1010D0000, 0x48000) = 0 0

1131/0x2e84: munmap(0x101118000, 0x4C000) = 0 0

1131/0x2e84: crossarch_trap(0x0, 0x0, 0x0) = -1 Err#45

1131/0x2e84: open(".\0", 0x100000, 0x0) = 4 0

1131/0x2e84: fcntl(0x4, 0x32, 0x16EDF30B8) = 0 0

1131/0x2e84: close(0x4) = 0 0

1131/0x2e84: fsgetpath(0x16EDF30C8, 0x400, 0x16EDF30A8) = 50 0

1131/0x2e84: fsgetpath(0x16EDF30D8, 0x400, 0x16EDF30B8) = 14 0

1131/0x2e84: csrctl(0x0, 0x16EDF34DC, 0x4) = -1 Err#1

1131/0x2e84: __mac_syscall(0x18CA1FC12, 0x2, 0x16EDF3420) = 0 0

1131/0x2e84: csrctl(0x0, 0x16EDF34CC, 0x4) = -1 Err#1

1131/0x2e84: __mac_syscall(0x18CA1CA45, 0x5A, 0x16EDF3460) = 0 0

1131/0x2e84: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16EDF29C8, 0x16EDF29C0, 0x18CA1E738, 0xD) = 0 0

1131/0x2e84: sysctl([CTL_KERN, 157, 0, 0, 0, 0] (2), 0x16EDF2A78, 0x16EDF2A70, 0x0, 0x0) = 0 0

1131/0x2e84: open("/\0", 0x20100000, 0x0) = 4 0

1131/0x2e84: openat(0x4, "System/Cryptexes/OS\0", 0x100000, 0x0) = 5 0

1131/0x2e84: dup(0x5, 0x0, 0x0) = 7 0

1131/0x2e84: fstatat64(0x5, 0x16EDF2551, 0x16EDF24C0) = 0 0

1131/0x2e84: openat(0x5, "System/Library/dyld/\0", 0x100000, 0x0) = 8 0

1131/0x2e84: fcntl(0x8, 0x32, 0x16EDF2550) = 0 0

1131/0x2e84: dup(0x8, 0x0, 0x0) = 9 0

1131/0x2e84: dup(0x7, 0x0, 0x0) = 10 0

1131/0x2e84: close(0x4) = 0 0

1131/0x2e84: close(0x7) = 0 0

1131/0x2e84: close(0x5) = 0 0

1131/0x2e84: close(0x8) = 0 0

1131/0x2e84: __mac_syscall(0x18CA1FC12, 0x2, 0x16EDF2F40) = 0 0

1131/0x2e84: shared_region_check_np(0x16EDF2B60, 0x0, 0x0) = 0 0

1131/0x2e84: fsgetpath(0x16EDF30E0, 0x400, 0x16EDF3008) = 82 0

1131/0x2e84: fcntl(0xA, 0x32, 0x16EDF30E0) = 0 0

1131/0x2e84: close(0xA) = 0 0

1131/0x2e84: close(0x9) = 0 0

1131/0x2e84: getfsstat64(0x0, 0x0, 0x2) = 11 0

1131/0x2e84: getfsstat64(0x101008050, 0x5D28, 0x2) = 11 0

1131/0x2e84: getattrlist("/\0", 0x16EDF3020, 0x16EDF2F90) = 0 0

1131/0x2e84: stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld_shared_cache_arm64e\0", 0x16EDF3380, 0x0) = 0 0

dtrace: error on enabled probe ID 1696 (ID 845: syscall::stat64:return): invalid address (0x0) in action #12 at DIF offset 12

1131/0x2e84: stat64("/Users/migermanenko/Documents/C/OC/Lab1/src/child\0", 0x16EDF2830, 0x0) = 0 0

1131/0x2e84: open("/Users/migermanenko/Documents/C/OC/Lab1/src/child\0", 0x0, 0x0) = 4 0

1131/0x2e84: mmap(0x0, 0x83E8, 0x1, 0x40002, 0x4, 0x0) = 0x101008000 0

1131/0x2e84: fcntl(0x4, 0x32, 0x16EDF2948) = 0 0

1131/0x2e84: close(0x4) = 0 0

1131/0x2e84: munmap(0x101008000, 0x83E8) = 0 0

1131/0x2e84: stat64("/Users/migermanenko/Documents/C/OC/Lab1/src/child\0", 0x16EDF2DA0, 0x0) = 0 0

1131/0x2e84: stat64("/usr/lib/libSystem.B.dylib\0", 0x16EDF1CF0, 0x0) = -1 Err#2

1131/0x2e84: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16EDF1CA0, 0x0) = -1 Err#2

1131/0x2e84: open("/Users/migermanenko/Documents/C/OC/Lab1/src/child\0", 0x0, 0x0) = 4 0

1131/0x2e84: __mac_syscall(0x18CA1FC12, 0x2, 0x16EDF03F0) = 0 0

1131/0x2e84: map_with_linking_np(0x16EDF02E0, 0x1, 0x16EDF0310) = 0 0

1131/0x2e84: close(0x4) = 0 0

1131/0x2e84: mprotect(0x101000000, 0x4000, 0x1) = 0 0

1131/0x2e84: open("/dev/dtracehelper\0", 0x2, 0x0) = 4 0

1131/0x2e84: ioctl(0x4, 0x80086804, 0x16EDEF9A8) = 0 0

1131/0x2e84: close(0x4) = 0 0

1131/0x2e84: shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

1131/0x2e84: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2

1131/0x2e84: bsdthread_register(0x18CD220F4, 0x18CD220E8, 0x4000) = 1073746399 0

1131/0x2e84: getpid(0x0, 0x0, 0x0) = 1131 0

1131/0x2e84: shm_open(0x18CBB9F41, 0x0, 0xFFFFFFFF9A748000) = 4 0

1131/0x2e84: fstat64(0x4, 0x16EDF0020, 0x0) = 0 0

1131/0x2e84: mmap(0x0, 0x8000, 0x1, 0x40001, 0x4, 0x0) = 0x101010000 0

1131/0x2e84: close(0x4) = 0 0

1131/0x2e84: csops(0x46B, 0x0, 0x16EDF015C) = 0 0

1131/0x2e84: ioctl(0x2, 0x4004667A, 0x16EDF00CC) = 0 0

1131/0x2e84: mprotect(0x101020000, 0x4000, 0x0) = 0 0

1131/0x2e84: mprotect(0x10102C000, 0x4000, 0x0) = 0 0

1131/0x2e84: mprotect(0x101030000, 0x4000, 0x0) = 0 0

1131/0x2e84: mprotect(0x10103C000, 0x4000, 0x0) = 0 0

1131/0x2e84: mprotect(0x101040000, 0x4000, 0x0) = 0 0

1131/0x2e84: mprotect(0x10104C000, 0x4000, 0x0) = 0 0

1131/0x2e84: mprotect(0x101018000, 0xC8, 0x1) = 0 0

1131/0x2e84: mprotect(0x101018000, 0xC8, 0x3) = 0 0

1131/0x2e84: mprotect(0x101018000, 0xC8, 0x1) = 0 0

1131/0x2e84: mprotect(0x101050000, 0x4000, 0x1) = 0 0

1131/0x2e84: mprotect(0x101054000, 0xC8, 0x1) = 0 0

1131/0x2e84: mprotect(0x101054000, 0xC8, 0x3) = 0 0

1131/0x2e84: mprotect(0x101054000, 0xC8, 0x1) = 0 0

1131/0x2e84: mprotect(0x101018000, 0xC8, 0x3) = 0 0

1131/0x2e84: mprotect(0x101018000, 0xC8, 0x1) = 0 0

1131/0x2e84: mprotect(0x101050000, 0x4000, 0x3) = 0 0

1131/0x2e84: mprotect(0x101050000, 0x4000, 0x1) = 0 0

1131/0x2e84: issetugid(0x0, 0x0, 0x0) = 0 0

1131/0x2e84: getentropy(0x16EDEF728, 0x20, 0x0) = 0 0

1131/0x2e84: getattrlist("/Users/migermanenko/Documents/C/OC/Lab1/src/child\0", 0x16EDEFFC0, 0x16EDEFFDC) = 0 0

1131/0x2e84: access("/Users/migermanenko/Documents/C/OC/Lab1/src\0", 0x4, 0x0) = 0 0

1131/0x2e84: open("/Users/migermanenko/Documents/C/OC/Lab1/src\0", 0x0, 0x0) = 4 0

1131/0x2e84: fstat64(0x4, 0x150604500, 0x0) = 0 0

1131/0x2e84: csrctl(0x0, 0x16EDF01AC, 0x4) = 0 0

1131/0x2e84: fcntl(0x4, 0x32, 0x16EDEFEA8) = 0 0

1131/0x2e84: close(0x4) = 0 0

1131/0x2e84: open("/Users/migermanenko/Documents/C/OC/Lab1/src/Info.plist\0", 0x0, 0x0) = -1 Err#2

1131/0x2e84: proc_info(0x2, 0x46B, 0xD) = 64 0

1131/0x2e84: csops_audittoken(0x46B, 0x10, 0x16EDF0230) = 0 0

1131/0x2e84: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16EDF0588, 0x16EDF0580, 0x19044CD3A, 0x15) = 0 0

1131/0x2e84: sysctl([CTL_KERN, 155, 0, 0, 0, 0] (2), 0x16EDF0618, 0x16EDF0610, 0x0, 0x0) = 0 0

1131/0x2e84: open("test.txt\0", 0x601, 0x1B6) = 4 0

24

Введите число: 1116/0x2dc1: read(0x0, "24\n\0", 0x80) = 3 0

1116/0x2dc1: write(0x4, "\030\0", 0x4) = 4 0

1131/0x2e84: read(0x0, "\030\0", 0x4) = 4 0

1131/0x2e84: write(0x4, "24\0", 0x2) = 2 0

1131/0x2e84: write(0x4, " \0", 0x1) = 1 0

1131/0x2e84: write(0x1, "\0", 0x4) = 4 0

1116/0x2dc1: read(0x5, "\0", 0x4) = 4 0

1116/0x2dc1: write(0x1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\207\320\270\321\201\320\273\320\276: \0", 0x1C) = 28 0

1116/0x2dc1: fsync(0x1, 0x0, 0x0) = 0 0

16

Введите число: 1116/0x2dc1: read(0x0, "16\n\0", 0x80) = 3 0

1116/0x2dc1: write(0x4, "\020\0", 0x4) = 4 0

1131/0x2e84: read(0x0, "\020\0", 0x4) = 4 0

1131/0x2e84: write(0x4, "16\0", 0x2) = 2 0

1131/0x2e84: write(0x4, " \0", 0x1) = 1 0

1131/0x2e84: write(0x1, "\0", 0x4) = 4 0

1116/0x2dc1: read(0x5, "\0", 0x4) = 4 0

1116/0x2dc1: write(0x1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\207\320\270\321\201\320\273\320\276: \0", 0x1C) = 28 0

1116/0x2dc1: fsync(0x1, 0x0, 0x0) = 0 0

-40

Программа завершена.

1116/0x2dc1: read(0x0, "-40\n\0", 0x80) = 4 0

1116/0x2dc1: write(0x4, "\330\377\377\377\0", 0x4) = 4 0

1131/0x2e84: read(0x0, "\330\377\377\377\0", 0x4) = 4 0

1131/0x2e84: write(0x1, "\001\0", 0x4) = 4 0

1116/0x2dc1: read(0x5, "\001\0", 0x4) = 4 0

1116/0x2dc1: write(0x1,
"\320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260
\320\267\320\260\320\262\320\265\321\200\321\210\320\265\320\275\320\260.\n\0", 0x28)
= 40 0

1116/0x2dc1: close(0x4) = 0 0

1116/0x2dc1: close(0x5) = 0 0

1131/0x2e84: close(0x4) = 0 0

1116/0x2dc1: wait4(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 1131 0

Вывод

С помощью системных вызовов и возможностей, которые дает язык Си можно организовывать взаимодействие меджу процессами, связывать их, передавать данные, файлы, а также обрабатывать их. Основные системные вызовы, которые используются для этого: fork(), pipe(), dup2(), read(), write(), open().